PERANCANGAN KANAL KOMUNIKASI PADA *TRANSACTION LEVEL MODELING* DALAM PERANCANGAN *EMBEDDED SYSTEM*

Maman Abdurohman[1], Kuspriyanto[2], Sarwono Sutikno[2], dan Arif Sasongko[2]

[1]Fakultas Informatika, IT Telkom, Jl. Telekomunikasi Terusan Buah Batu, Bandung, 40257, Indonesia
[2]STEI, Institut Teknologi Bandung, Jl. Ganesha 10, Bandung, 40132, Indonesia

E-mail: m_abdurohman@yahoo.com

Abstrak

Pada *embedded system* terdapat dua bagian penting yaitu komponen komputasi (*register*) dan komponen komunikasi. Komponen komunikasi menjadi perhatian penting pada mekanisme pemodelan level transaksi (*Transaction Level Modeling, TLM*). Kanal komunikasi adalah komponen untuk transaksi antar *register*. Fokus pembahasan TLM adalah perancangan kanal yang dapat mengakomodasi untuk peningkatan level transaksi. Kanal (*channel*) adalah implementasi *bus* untuk komunikasi antar komponen pada *embedded system*. Hal ini adalah kunci penting untuk mencapai impelementasi TLM untuk meningkatkan efisiensi pemodelan. Pada *paper* ini diusulkan beberapa definisi rancangan kanal sebagai implementasi TLM untuk perancangan *embedded system*. Hasilnya menunjukan bahwa rancangan kanal dapat berjalan sebagai *bus* untuk transaksi pada TLM. *Paper* ini menggunakan SystemC sebagai bahasa pemodelan.

Kata Kunci: *bus, channel, systemC, TLM*

Abstract

On embedded systems, there are two important parts: computational components (registers) and communication components. Communication component becomes an important attention on the mechanism of transaction level modeling (TLM). Communication channel is a component for transactions between registers. The focus of TLM is the design of the channel that could accommodate for the increased level of transactions. Channel is the implementation of the bus for communication between components in embedded systems. This is an important key to achieve the implementation of TLM to improve the efficiency of modeling. This paper proposed a definition of the channel design as the implementation of TLM for embedded systems design. The result shows that the design of the channel can run as a bus for transactions on the TLM. This paper uses SystemC as modeling language.

Keywords: *bus, channel, systemC, TLM*

1. Pendahuluan

Hardware dan *software* adalah dua perangkat yang penting dalam perancangan dan implementasi *embedded system*. Keduanya merupakan aspek yang saling melengkapi dalam implementasi *embedded system*. Pada tahap awal dilakukan pendefinisian perangkat keras kemudian perangkat lunak dan aplikasi. Semakin cepat proses perancangan *embedded system* akan semakin baik untuk meningkatkan efisiensi.

Proses perancangan *embedded system* dimulai dengan spesifikasi produk, pemisahan HW/SW (*Hardware/Software*), perancangan HW/SW secara rinci, integrasi dan pengujian, seperti pada gambar 1. Biasanya ada dua metrik yang digunakan untuk menentukan kesesuaian hasil rancangan yaitu biaya dan kecepatan. Kedua metrik tersebut banyak digunakan dalam proses perancangan. Metrik lainnya seperti ukuran, konsumsi listrik, fleksibilitas, skalabilitas, dan lain-lain.

Spesifikasi produk menjelaskan batasan-batasan perancangan sesuai dengan keinginan perancang. Proses pemisahan HW/SW melakukan pemisahan antara bagian sistem yang akan diimplementasikan dalam bentuk HW dan SW. Perancangan rinci HW dan SW dilakukan secara paralel. Kemudian proses integrasi dan pengujian.

Perancangan *level register* dilakukan dengan menggunakan bahasa pemrograman perangkat keras, yang umum digunakan adalah Verilog, VHDL, dan ESTEREL. Proses verifikasi dan

pengujian dilakukan untuk memastikan kesesuaian hasil rancangan dengan spesifikasi awal.

2. Metodologi

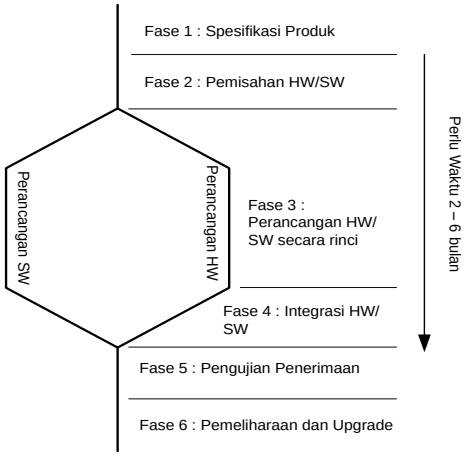
Abstraksi perancangan adalah bagaimana cara memandang objek rancangan. Dalam perancangan *embedded system* dimulai dari perancang yang paling rendah yaitu *level transistor*, gerbang – *flip-flop*, *register* dan elektronik. Perancangan *level transistor* dan gerbang sudah lama ditinggalkan karena proses verifikasinya yang sangat lambat dan tidak memadai lagi untuk memenuhi tuntutan kecepatan perancangan. Dengan kemampatan IC lebih dari lima miliar per *chip*, perancangan berbasis gerbang – *flip-flop* tidak lagi mencukupi terutama dalam hal kecepatan proses.

Peningkatan level abstraksi perancangan telah dilakukan yaitu dengan meningkatkan level abstraksi perancangan dari level gerbang ke *Register Transfer Level* (RTL). Langkah ini sebenarnya termasuk langkah yang signifikan dan ada yang menyebutnya sebagai revolusi teknologi [1] bagi peningkatan produktivitas perancang. Perancang tidak lagi disibukkan untuk membahas pada level gerbang. Perancangan level gerbang sudah dianggap mafhum dan tidak perlu lagi dibahas. Perancangan dimulai dengan mendefinisikan komponen-komponen di atas gerbang yaitu *register*.

Terdapat dua bahasa perancangan level RTL yang dikenal luas dan telah distandarkan oleh IEEE yaitu Verilog dan VHDL. Keduanya, secara *de facto*, dianggap sebagai bahasa perancangan perangkat keras, *Hardware Description Language* (HDL). Pada awalnya, kedua bahasa ini adalah alat yang digunakan untuk menyimpan notasi hasil rancangan, bukan bahasa pemrograman umumnya.

Level abstraksi RTL dianggap belum menjawab tuntutan perkembangan *embedded system*. Beberapa kondisi yang layak dipertimbangkan di sini antara lain, pertama semakin mampatnya teknologi IC menuntut kemampuan untuk meletakkan miliaran transistor pada suatu *chip* yang kecil. Kedua, aplikasi semakin kompleks dan terus semakin kompleks dan terintegrasi dengan komunikasi, kendali, sistem *mobile*, dan menyebar ke mana-mana. Ketiga, fungsionalitas perangkat keras dan perangkat lunak semakin fleksibel untuk dipertukarkan. Semua fungsi bisa diperangkatkeraskan dan diperangkatlunakkan. Sistem perancangan dituntut untuk dapat melakukan pemisahan yang tepat sesuai dengan kebutuhan perangkat.

Pada dua dekade terakhir (awal tahun 1990 sampai sekarang) pemikiran ke arah peningkatan level abstraksi perancangan mulai banyak dibicarakan. Seharusnya ada sesuatu level abstraksi di atas RTL. Ide awal pengistilahan level perancangan di atas RTL adalah level elektronik, dengan istilah *Electronic System Level* (ESL). Walaupun sampai sekarang istilah ini masih belum standar, dalam arti belum ada *standard* yang jelas mengenai ruang lingkup dan proses-proses apa saja yang ada pada level tersebut. Sejauh ini definisi yang dikemukakan tentang ESL adalah "Suatu level di atas RTL termasuk di dalamnya perancangan perangkat keras dan perangkat lunak" [2]. Masih sangat generik. Namun demikian, secara garis besar metode perancangan apapun namanya dan di manapun levelnya harus tetap memiliki fungsi-fungsi pemodelan, simulasi, validasi, dan verifikasi. Masih terbukanya pembahasan abstraksi level ESL memberikan keleluasaan kepada para peneliti untuk memberikan istilah disesuaikan dengan ruang lingkup dan target penelitiannya. Terdapat istilah-istilah yang banyak digunakan oleh para peneliti dalam menyebut perancangan level ESL, diantaranya, *hardware/software co-design model*, *architectural model*, RTL, dan *software model*, *cell-level model*, *high level synthesis*, serta *system level synthesis*.



Gambar 1. Alur proses perancangan *embedded system* secara umum.

Istilah-istilah ini akan terus bertambah sebelum adanya suatu kesepakatan bersama tentang istilah yang lebih tepatnya atau ditetapkan oleh badan standardisasi internasional. Hal ini tidak menjadi hambatan, yang penting ide meningkatnya level abstraksi perancangan telah mulai menunjukan hasil yang bisa dirasakan.

SystemC adalah *library* C++ yang menyediakan berbagai komponen untuk digunakan pada perancangan level transaksi. Kemampuan yang dimiliki SystemC adalah pemrograman sekuensial sebagaimana C++ dan pemrograman *concurrent*. Kemampuan pemrograman *concurrent* ini memungkinkan SystemC untuk digunakan dalam memodelkan komponen perangkat keras dan perangkat lunak yang kompleks. Dibandingkan dengan VHDL dan Verilog, SystemC memiliki kelebihan dalam hal pemrograman terstruktur yang tidak dimiliki oleh VHDL dan Verilog. Dalam hal ini, SystemC memiliki fleksibilitas yang lebih tinggi sebagaimana halnya bahasa C++. Kelebihan lain adalah sifat alami SystemC sebagai bahasa C++, sehingga spesifikasi perangkat keras yang ditulis dalam bahasa C tidak perlu lagi diubah ke dalam bentuk lain [3].

Library SystemC digunakan untuk mendukung pemodelan *level* sistem. SystemC mendukung berbagai level abstraksi dan dapat digunakan untuk perancangan dan verifikasi yang cepat dan efisien. *Library* SystemC disediakan oleh *Open SystemC Initiative* (OSCI), sebuah organisasi nirlaba yang terbuka. OSCI didukung oleh sejumlah perusahaan, universitas dan individu yang tertarik pada pengembangan pemodelan dengan abstraksi yang lebih tinggi. *Library* SystemC dapat diperoleh secara gratis di www.systemc.org.

Pendefinisan SystemC berada di atas bahasa C++. Terdapat beberapa bahasa inti dan tipe data yang terdapat dalam SystemC. Semuanya dibangun di atas bahasa C++. Bahasa inti terdiri dari *Module/Process*, *Port/Interface*, *Event*, *Channel* dan *Event-driven simulation kernel* (gambar 2).

Tipe data yang telah didefinisikan terdiri dari *4-valued logic types* (01XZ), *bit/logic vector*, *arbitrary precision integer*, *fixed point* dan tipe-tipe lain yang didefinisikan langsung oleh pengguna berdasarkan bahasa C++. Pada lapis di atasnya terdapat *elementary channel* seperti *signal*, *timer*, *mutex*, *semaphore* dan FIFO yang menyediakan mekanisme komunikasi antar objek secara *concurrent*.

Module adalah kelas C++ yang membungkus objek perangkat keras atau perangkat lunak. *Module* ini didefinisikan dalam SystemC sebagai kelas sc_module. *Module* ini setara dengan *entity/architecture* pada Verilog atau VHDL, yang mewakili suatu blok dasar komponen. Suatu *module* dengan *module* lain berkomunikasi melalui *channel* dan *port*. Dalam *module* terdapat sejumlah proses *concurrent* sebagai implementasi perilaku komponen tersebut.

Port adalah objek yang terdapat dalam *module*, yang berfungsi menghubungkan *module* dengan dunia luar. *Port-port* yang telah didefinisikan dalam SystemC adalah sc_in<>, sc_out<>, sc_inout<>, sc_fifo_in<>, sc_fifo_out<>, dan lain-lain.

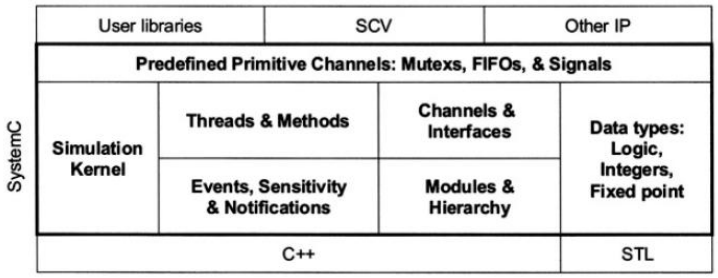
Terdapat dua jenis proses yaitu: SC_METHOD dan SC_THREAD. Keduanya mirip, hanya berbeda dalam masalah SC_METHOD tidak dapat dihentikan sementara pada saat eksekusi sedangkan SC_THREAD dapat dihentikan sementara pada saat eksekusi. Proses kompilasi SystemC dapat dilihat pada gambar 3.

Channel adalah media komunikasi SystemC. Sifatnya lebih umum dibandingkan sinyal. Beberapa *channel* dalam SystemC seperti sc_signal, sc_fifo, sc_semaphore. Prinsip komposisi sama dengan perancangan hierarki. Pada sistem yang kompleks, komposisi sangat diperlukan.

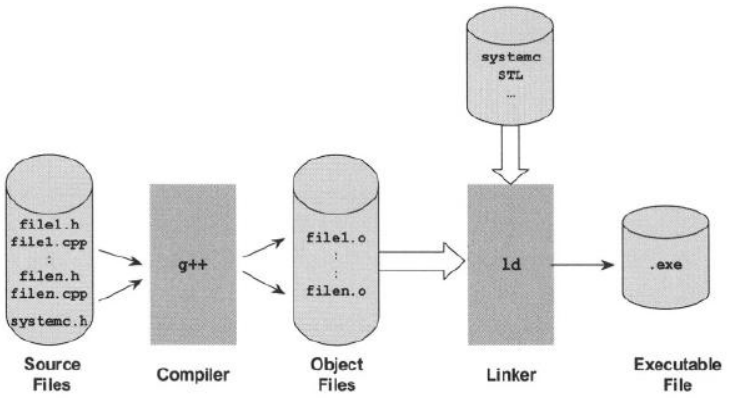
TLM adalah sebuah abstraksi pemodelan baru yaitu metodologi untuk memodelkan perangkat keras saat ini menuju pada level abstraksi yang lebih tinggi. Pemodelan yang dilakukan saat ini adalah pemodelan level RTL (*Register Transfer Level*). Pada level ini komponen perangkat keras beserta sinyal pewaktuan yang akurat. Pemodelan seperti ini masih cocok untuk sistem yang sederhana. Sistem yang kompleks membutuhkan waktu yang cukup lama untuk dievaluasi dengan model abstraksi RTL.

Pada pemodelan level transaksi tidak dilakukan pendefinisian sinyal secara tunggal melainkan sekumpulan sinyal yang beroperasi pada tipe data abstrak, yang dapat meningkatkan kecepatan waktu simulasi. Dengan cara ini proses simulasi pada level transaksi jauh lebih cepat dibandingkan dengan pemodelan level RLT.

Konsep utama yang dikenalkan dalam TLM adalah pemisahan komunikasi antar komponen dari proses yang ada dalam komponen. Komunikasi dimodelkan sebagai *channel* yang dirancang sebagai satu abstraksi tingkat tinggi yang tidak menunjukkan secara rinci proses yang ada di dalamnya. Dengan cara ini, TLM dapat meningkatkan kecepatan dalam proses simulasi perancangan dan dapat mengeksplorasi dan validasi rancangan pada level abstraksi yang lebih tinggi.



Gambar 2. Arsitektur SystemC [2].



Gambar 3. Proses kompilasi SystemC [4].

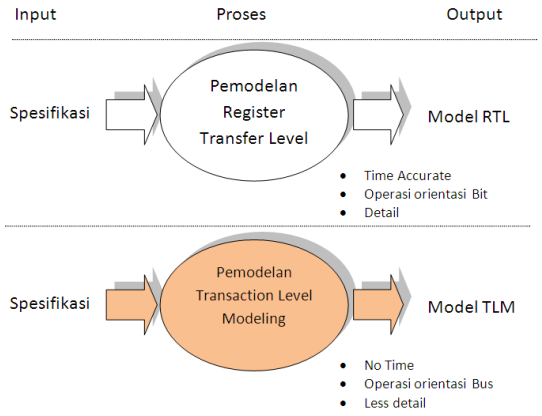
Model Algoritma
Model UTF (UnTimed Functional)
Model TF (Timed Functional)
Model BCA (Bus Cycle Accurate)
Model CA (Cycle Accurate)
Model RTL (*Register* Transfer Level)

Gambar 4. Pemodelan level transaksi.

Meskipun tidak secara akurat menghasilkan besaran-besaran parameter seperti pada level RTL. Hal-hal yang dilakukan pada masing-masing level antara lain, pertama level algoritma dan UTF untuk verifikasi fungsional dan validasi algoritma. Kedua, level TF dan BCA untuk *benchmark* kasar, pembangunan perangkat lunak aplikasi dan analisis arsitektur. Ketiga, level CA dan RTL untuk *benchmark* rinci, pembangunan *driver* dan analisis mikroarsitektur (gambar 4).

Pemodelan level transaksi (gambar 5) berada pada level abstraksi tingkat yang lebih tinggi. Menghilangkan rincian *cycle accurate* dan *timed funcional*. Fungsi-fungsi yang didefinisikan pada level yang lebih atas dengan mengabaikan rincian pewaktuan memungkinkan untuk pemodelan yang lebih cepat.

Pada pemodelan TLM keluaran yang dihasilkan berupa model dengan karakteristik umum tidak menggunakan pewaktuan yang akurat, waktu yang digunakan adalah waktu simulasi sehingga proses pemodelan bisa lebih cepat. Karakteristik kedua adalah operasi alaminya dalam bertransaksi berorientasi *bus*. Suatu *bus* dipandang secara utuh dan tidak memilahnya dalam *bit* per *bit*. Ketiga adalah tidak rinci dalam artian proses tidak melakukan operasi *bit* per *bit*. Ketiga sifat dasar TLM ini yang memungkinkan proses pada TLM lebih cepat.

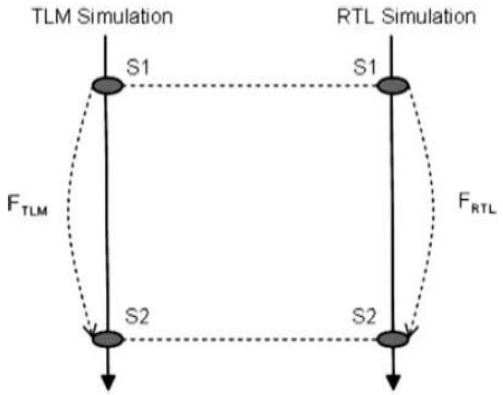


Gambar 5. Pemodelan RTL dan TLM.

Pemodelan level transaksi tidak lepas dari verifikasi untuk tetap menghubungkannya dengan pemodelan level RTL. Diperlukan adanya sinkronisasi antara pemodelan level transaksi dengan RTL sehingga fungsionalitas sistem tetap berjalan sesuai dengan spesifikasi.

Sinkronisasi ini dilakukan sesuai dengan kebutuhan. Terlalu banyak titik-titik sinkronisasi antara kedua model akan mengaburkan arti pentingnya pemodelan level transaksi karena akan mendekati model RTL. Pemodelan level transksi akan terlalu dekat dengan pemodelan level RTL atau *cycle accurate*. Kecepatan yang lebih tinggi sulit dicapai. Demikian juga jika terlalu sedikit sinkronisasi yang dilakukan, maka pemodelan level sistem tidak dapat memenuhi spesifikasi yang telah ditentukan.

Evolusi sistem dari mulai kondisi S1 dan S2 ditunjukan sebagai Ftlm dan Frtl. S1 dan S2 adalah dua titik pengamatan yang dilakukan pada kedua simulasi (gambar 6).



Gambar 6. Simulasi TLM vs RTL [5].

Kanal dalam SystemC adalah media komunikasi. Kanal lebih umum daripada sinyal. Beberapa tipe kanal dalam SystemC adalah sc_signal, sc_fifo, sc_semaphore, dan lain-lain. Komposisi masing-masing kanal mirip dengan perancangan kanal hierarki. Pada sistem yang kompleks dibutuhkan adanya komposisi dalam kanal komunikasi. Pada *paper* ini dirancang kanal komunikasi yang diperlukan untuk implementasi konsep TLM.

Kanal merupakan bagian penting dalam sebuah TLM. Beberapa kanal yang didefinisikan antara lain, pertama, FIFO (Kanal dengan sifat *First In First Out)* yang merupakan kanal untuk mentransfer data dari sumber ke tujuan dengan menggunakan kanal dengan disiplin FIFO. Data yang masuk terlebih dahulu akan dilayani lebih dahulu.

Kedua, FIFO *Wrapper* merupakan modifikasi dari kanal FIFO. FIFO *Wrapper* merupakan kanal FIFO dengan tambahan aturan untuk meningkatkan kemampuan kanal. Terdapat beberapa kelas yang ditambahkan ke dalam rancangan kanal FIFO. FIFO *Wrapper* lebih cocok untuk transfer data. Terdapat sebuah *arbiter* yang berfungsi untuk mengatur proses transaksi.

Ketiga, g*eneral bus* merupakan kanal sistem *bus* yang mensimulasikan dengan lima komponen aktif dan satu *arbiter*. Keempat, OCP (*Open Core Protocol*, tl1 : Level 1) merupakan kanal OCP dibangun untuk mengimplementasikan standar protokol OCP. OCP level 1 merupakan *standard* paling rinci. Kelima, OCP tl2 : Level 2 sama dengan OCP level 1 tapi dengan abstraksi yang lebih tinggi dan tidak terlalu rinci.

Perancangan kanal komunikasi untuk TLM antara lain, FIFO (*First In First Out*) *channels, FIFO wrapper, general bus,* OCP (*Open Core Protocol*) Level 1, OCP (*Open Core Protocol*) Level 2. FIFO (*First In First Out*) *channels* menggunakan proses *producer* - *consumer* untuk mengimplementasikan rancangan ini. Kanal FIFO digunakan untuk menghubungkan antara komponen *producer* dan *consumer*.

Gambar 7. Kanal FIFO.

Gambar 7 menunjukkan koneksi antara *producer* dan *consumer*. *Producer* adalah komponen yang mengirimkan data ke *consumer*. Kanal FIFO melayani sebagai *buffer* untuk data yang dikirim ke *consumer*. Ukuran *buffer* tergantung dari definisi pengguna. Pada kanal ini didefinsikan tiga kelas utama yaitu: *producer,*

consumer dan *top*. Kelas *top* adalah kelas yang menyatukan seluruh proses dalam sistem. Pada bahasa pemrograman FIFO didefinisikan pada *producer* dan *consumer* dengan tipe yang berbeda. Pada kelas *producer* :

```
sc_port<sc_fifo_out_if<char>>out;
```

Gambar 8. Kelas *producer.*

Pada kelas *consumer* :

```
sc_port<sc_fifo_in_if<char>>i;
```

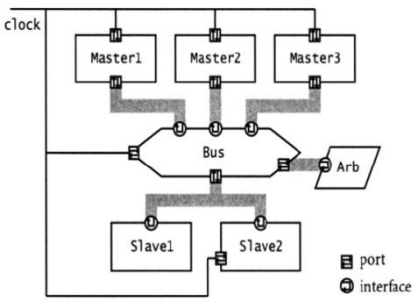
Gambar 9. Kelas *consumer.*

```
template <class T> class hw_fifo:public sc_module
template <class T> class hw_fifo_wrapper : public sc_module,
public sc_fifo_in_if<T>,
public sc_fifo_out_if<T>
   { … };
```

Gambar 10. FIFO *wrapper.*

FIFO *wrapper* dirancang sama dengan kanal FIFO dengan tambahan komponen *arbiter*. Pada perancangan pemodelan ditambahkan sebuah kelas "hw_fifo" dan kelas *wrapper* untuk mengimplementasikan FIFO *wrapper*. Hal ini dapat dilihat pada gambar 10.

Pada *general bus* didefinisikan tujuh komponen: Master1, Master2, Master3, Slave1, Slave2, *Arbiter* dan *Bus* sendiri. *Bus* menghubungkan semua komponen dalam sistem (gambar 11).



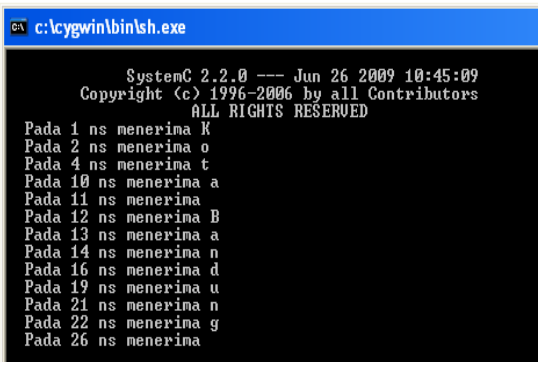
Gambar 11. Diagram blok *general bus* [6].

OCP (*Open Core Protocol*) level 1 merupakan implementasi dari standar OCP. Model OCP dirancang untuk memudahkan untuk menggunakan serta mengelola fungsionalitasnya. Pada rancangan OCP terdapat dua komponen utama yaitu simpleMaster dan SimpleSlave yang berkomunikasi menggunakan kanal OCP. Level 1 menunjukan level rinci dari kanal. OCP (*Open Core Protocol*) Level 2 sama dengan OCP level 1 dengan tingkat kerincian yang lebih rendah. Kanal ini dirancang untuk meningkatkan waktu proses.

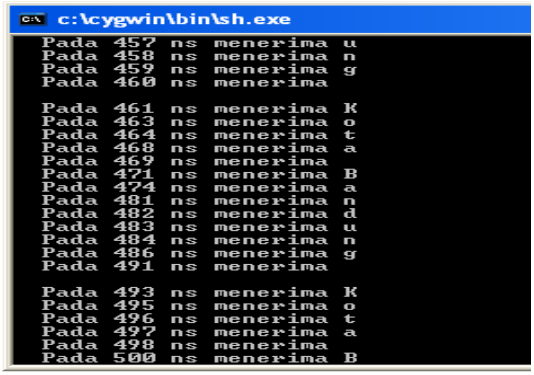
3. Hasil dan Pembahasan

Dalam pengujian kanal-kanal yang telah didefinisikan digunakan pemrograman C++ yang dijalankan di atas Cygwin dengan IDE Netbeans 6.1. Pada pengujian FIFO (*First In First Out*) dan FIFO *wrapper* dirancang proses pengiriman dari *consumer* sebuah teks "Kota Bandung" ke *consumer* karakter demi karakter.

Pada saat proses pengiriman akan diamati setiap karakter yang dikirim dan diterima oleh tujuan. Gambar 12 menunjukkan bahwa data dapat dikirim dari *producer* ke *consumer* melalui kanal FIFO dan FIFO_*Wrapper*. Hal ini menunjukkan bahwa kanal dapat digunakan pada TLM.



(a)

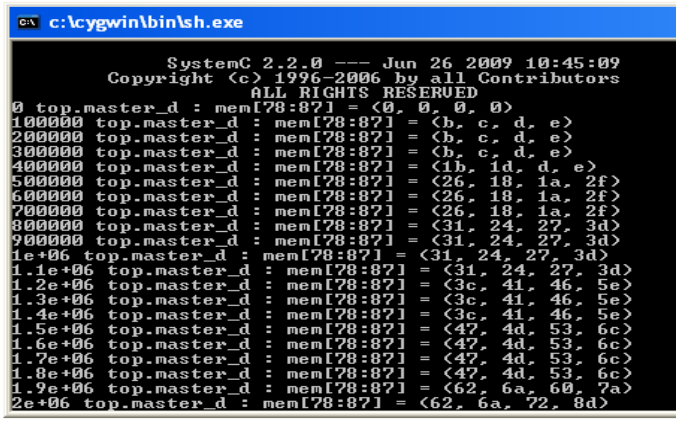


(b)

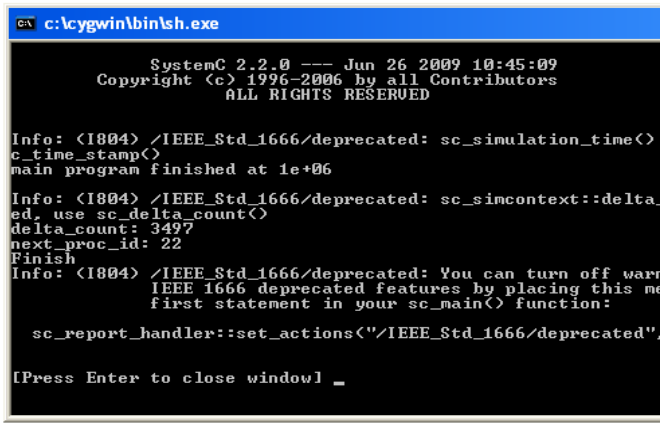
Gambar 12. (a) hasil kompilasi kanal FIFO dan (b) kanal fifo_wrapper.

Pada *general bus* dirancang tiga komponen *master* yang aktif yang mengirim data ke *slave.* Penjelasan ini dapat dilihat pada gambar 13. Hasil

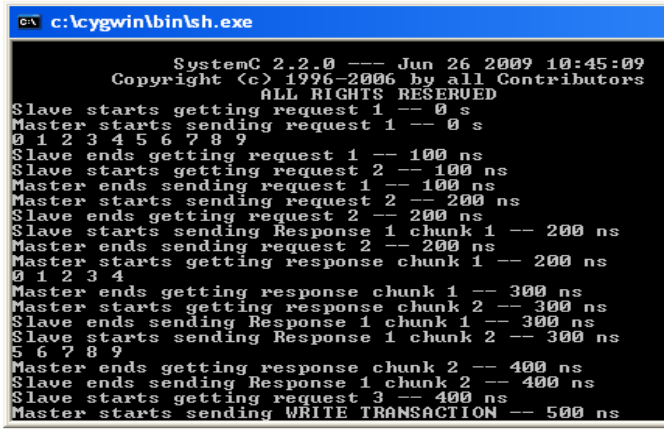
dari *compile general bus* menunjukkan bahwa kanal dapat mengirim data dari komponen aktif dan sebaliknya (gambar 12). OCP TL1 dan TL2 dirancang sebagai kanal dengan standar OCP (*Open Core Protocol*). Hasilnya menunjukkan bahwa data dapat di-transfer dari *master* ke *slave* menggunakan kanal tersebut (gambar 14).



Gambar 13. Hasil kompilasi kanal general_bus.



(a)



(b)

Gambar 14. (a) hasil kompilasi kanal OCP tl1 dan (b) kanal tl2.

4. Kesimpulan

Hasil penelitian menunjukkan bahwa kanal-kanal yang dirancang dapat digunakan untuk transfer data dari sumber ke tujuan dengan model TLM. Proses ini memerlukan waktu yang lebih sedikit untuk membangun prototipe. Kita dapat melakukan verifikasi lebih awal. Dengan verifikasi lebih awal ini dapat meningkatkan kecepatan proses perancangan *embedded system*.

Ucapan Terima Kasih

Maman Abdurohman mengucapkan terima kasih kepada Fakultas Informatika – IT Telkom dan Fakultas STEI ITB atas dukungan finansial dan perangkat penelitian sehingga penelitian ini bisa diselesaikan.

Referensi

[1] F. Vahid & T. Givargis, *Embedded System Design: A Unified Hardware/Software Introduction*, John Wiley & Sons, Inc., New York, 2002.

[2] K.S. Chatha, "System-Level Cosynthesis of Transformative Application for Heterogeneous Hardware-Software Architecture," Doctoral Dissertation, University of Cincinnati, United States, 2001.

[3] D.A. Mathaikutty, "Metamodeling Driven IP Reuse for System-on-chip Integration and Microprocessor Design," Doctoral Dissertation, Virginia Polytechnic Institute and State University, United States, 2007.

[4] D. Black & J. Donovan, *SystemC : From the Ground Up*, Kluwer Akademic Publisher, USA, 2004.

[5] F. Ghenassia, *Transaction Level Modeling with SystemC : TLM concepts and application for Embedded System*, Springer Inc., Netherlands, 2005.

[6] G.E. Moore, "Cramming more components onto integrated circuits" *In Proceedings of the IEEE*, pp. 82-85, 1998.